Proceedings Paper:
Bai, L., Cui, Lixin, Escolano, F. et al. (1 more author) (2016) An edge-based matching kernel on commute-time spanning trees. In: Davis, L., Bimbo, A. Del and Lovell, B., (eds.) 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE Computer Society , Los Alamitos, CA, USA , pp. 2103-2108.

https://doi.org/10.1109/ICPR.2016.7899946

An Edge-based Matching Kernel on Commute-time Spanning Trees

Lu Bai*, Lixin Cui*, Francisco Escolano†, and Edwin R. Hancock‡
*School of Information, Central University of Finance and Economics, 39 South College Road, Beijing, China
†Department of Computer Science and Artificial Intelligence, University of Alicante, Spain
‡Department of Computer Science, University of York, York, YO10 5DD, UK

***Abstract*—Bai and Hancock recently proposed a novel edge-based matching kernel for graphs [1], by aligning depth-based representations. Unfortunately, one drawback arising in their kernel is the computational inefficiency for large graphs. This follows the fact that their kernel is essentially defined on directed line graphs. Moreover, the computational complexity of the kernel is cubic in the vertex number of the line graph. Since the directed line graph is a dual representation and each vertex represents a directed arc residing on the edge of the original graph. For a graph having n vertices, there may be at most $n(n-1)$ directed arcs residing on the edges and thus at most $n(n-1)$ vertices in the directed line graph. As a result, computing the kernel through the line graphs may require time complexity $O(n^6)$ for the worst case, making the kernel unapplicable for graphs having hundreds of vertices. The aim of this paper is to overcome this inefficiency, by proposing a new edge-based matching kernel. In order to cope with large graph structures, we propose to construct a sparser version of the original graph using the simplification method introduced in [2]. More specifically, we compute the minimum spanning tree over the commute time matrix of a graph. This spanning tree representation minimizes the number of edges of the original graph while preserving most of its structural information. With this simplification method to hand, the new edge-based matching kernel between two graphs is then computed on the directed line graphs transformed from their respective minimum spanning trees. We show that this strategy significantly reduces the computational complexity to $O(n^3)$. We evaluate the performance of the proposed kernels on several standard graph datasets. The experimental results demonstrate the effectiveness and efficiency.**

I. Introduction

In pattern recognition, graph kernels are powerful tools for structural analysis on graphs [3]. The advantages of using graph kernels are twofold. First, graph kernels can characterize a graph structure in a high dimensional space and thus better preserve the topology information. Second, graph kernels provide an elegant way of making the standard machine learning algorithms for vectors applicable for graph structures.

Most existing graph kernels fall into the instances of R-convolution [4], that is a generic way to define a graph kernel. For a pair of graphs, an R-convolution kernel is computed by decomposing each graph into small substructures and counting the pairs of isomorphic substructures. Generally speaking, a new type of decomposition usually results in a new graph kernel. Following this scenario, most existing graph kernels can be generally categorized into three classes [1], i.e. graph kernels based on comparing all pairs of a) walks [5], [6], b) paths [7], [8] and c) restricted subgraph or subtree structures [9], [10], [11]. Unfortunately, all the aforementioned R-convolution kernels suffer from the drawback of neglecting structural correspondence information. This follows the fact that all these kernels add an unit kernel value by roughly identifying a pair of isomorphic substructures, and cannot establish reliable structural correspondences between the substructures. This drawback limits the precise kernel-based similarity measure for graphs. To overcome the shortcoming, in our previous work [12], we have developed a new depth-based matching kernel for graphs. The depth-based matching kernel is based on aligning depth-based representations of vertices, and is computed by counting the number of matched vertex pairs. Moreover, this kernel can be seen as an aligned subgraph kernel that encapsulates location correspondence information between pairwise subgraphs. As a result, the depth-based matching kernel overcomes the shortcoming of neglecting location correspondences between substructures arising in R-convolution kernels.

To develop the depth-based matching kernel one step further, in [1] we have developed an edge-based matching kernel. More specifically, we transform each graph into a directed line graph [13] and compute the depth-based representations of vertices on the line graphs. The resulting edge-based matching kernel is computed by aligning the vertex representations. Since the directed line graph can expose the original graph in a high dimensional space, the edge-based matching kernel can reflect richer graph characteristics than the depth-based matching kernel on original graphs. Unfortunately, the edge-based matching kernel may suffer from computational inefficiency for large graphs. This is because the computational complexity of the edge-based matching kernel is cubic in the vertex number of the line graph. Since the directed line graph is a dual representation and each vertex represents a directed arc residing on the edge of the original graph [13]. For a graph having n vertices, there may be at most $n(n-1)$ directed arcs residing on the edges and thus at most $n(n-1)$ vertices in the directed line graph. As a result, computing the kernel through the line graphs may require time complexity $O(n^6)$ for the worst case, making the kernel unapplicable for graphs having hundreds of vertices.

The aim of this paper is to overcome this inefficiency, by proposing a new edge-based matching kernel. In order to cope with large graph structures, we propose to construct a sparser version of the original graph and reduce the number of the edges, through the commute time [2]. For a graph $G(V,E)$ and a pair of vertices $v,u \in V$, the commute time $CT(u,v)$ is the expected time for the random walk to travel from u to v and then return. Since the commute time can be

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 17, 2016.

anticipated to be a more robust measure of the proximity of data than the raw proximity matrix [2], the commute time is robust with respect to the structural noise of a graph, e.g., the edge deletion. As a result, the commute time provides us an elegant way of represent an ideal candidate to sparsify the original graph structure. More specifically, we compute the minimum spanning tree over the commute time matrix of an original graph [2]. This spanning tree representation minimizes the number of edges of the original graph while preserving most of its structural information, relying on the property of the commute time. With this simplification method to hand, the new edge-based matching kernel between two graphs is then computed on the directed line graphs transformed from their respective minimum spanning trees. We show that this strategy significantly reduces the computational complexity to $O(n^3)$. We evaluate the performance of the proposed kernels on several standard graph datasets. The experimental results demonstrate the effectiveness and efficiency.

The remainder of this paper is organized as follows. Section II reviews the concept of the original edge-based matching kernel developed in [1]. Section III defines the new edge-based matching kernel on the commute-time spanning trees. Section IV provides our experimental evaluation. Finally, Section V concludes our work.

II. The Edge-based Matching Kernel

In this section, we review the concept of the original edge-based matching kernel [1]. Moreover, we analyze the computational complexity of the edge-based matching kernel.

A. Directed Line Graphs

To compute the edge-based matching kernel, we need to transform each graph into a directed line graph. The reason of using the directed line graph is that it is a dual representation [13], and each vertex corresponds to a directed arc residing on the edge of the original graph, i.e., each vertex of the line graph can represent a corresponding edge in the original graph. As a result, the directed line graph provides a way of developing new edge-based matching kernels by aligning the vertices on the line graphs.

Based on the definition of Ren et al. in [13], for a sample graph $G(V,E)$, the directed line graph $OLG(V_L,E_{dL})$ is a dual representation of $G(V,E)$. To obtain $OLG(V_L,E_{dL})$, we first construct the associated symmetric digraph $SDG(V,E_d)$ of $G(V,E)$, by replacing every edge $e(u,w)\in E(G)$ by a pair of reverse arcs, i.e., directed edges $e_d(u,w)\in E_d(G)$ and $e_d(w,u)\in E_d(G)$ for $u,w\in V$. The directed line graph $OLG(V_L,E_{dL})$ is the directed graph with vertex set V_L and arc set E_{dL} defined as

$$\begin{array}{l} V_L \;\; = E_d(SDG), \\ E_{dL} = \{(e_d(u,v),e_d(v,w))\in E_d(SDG)\times E_d(SDG)\}, \end{array} \tag{1}$$

where $u,v,w\in V$ and $u\neq w$. The Perron-Frobenius operator $\boldsymbol{T}=[T_{i,j}]_{|V_L|\times|V_L|}$ of $G(V,E)$ is the adjacency matrix of the associated directed line graph $OLG(V_L,E_{dL})$. An example of transforming an original graph into a directed line graph is shown in Figure 1. Figure 1(a) shows the original graph and Figure 1(b) shows the associated symmetric digraph.

B. The Depth-based Representation of An Edge through the Directed Line Graph

Since each vertex of the directed line graph can represent a corresponding edge of the original graph. We proposed to compute the h-layer depth-based representation of the edge through the line graph, i.e., we compute the representation around a corresponding vertex of the line graph.

The h-layer undirected depth-based representation For a graph $G(V,E)$ and its directed line graph $G_D(V_D,\overrightarrow{E}_D))$, we commence by computing the h-layer undirected depth-based representation on the undirected line graph $G_U(V_U,E_U)$, that is computed by replacing each unidirectional edge of G_D into a bidirectional edge. For $G_U(V_U,E_U)$ and a vertex $v_U\in V_U$, let a vertex set $N_{v_U}^K$ be defined as $N_{v_U}^K=\{u_U\in V_U \mid S_G(v_U,u_U)\leq K\}$, where S_G is the shortest path matrix of G_U and $S_G(v_U,u_U)$ is the shortest path length between v_U and u_U. For G_U, the K-layer expansion subgraph $\mathcal{G}_{v_U}^K(\mathcal{V}_{v_U}^K;\mathcal{E}_{v_U}^K)$ around v_U is

$$\left\{\begin{array}{l} \mathcal{V}_{v_U}^K \;\; =\{u_U\in N_{v_U}^K\}; \\ \mathcal{E}_{v_U}^K =\{u_U,w_U\in N_{v_U}^K,\;(u_U,w_U)\in E_U\}. \end{array}\right. \tag{2}$$

For G_U, the h-layer undirected depth-based representation around $v_U\in V_U$ is

$$DB_{G_U}^h(v_U)=[H_S(\mathcal{G}_{v_U}^1),\cdots,H_S(\mathcal{G}_{v_U}^K),\cdots,H_S(\mathcal{G}_{v_U}^h)]^\top, \tag{3}$$

where $(K\leq h)$. $H_S(\mathcal{G}_{v_U}^K)$ is the Shannon entropy of $\mathcal{G}_{v_U}^K$ associated with the steady state random walk (SSRW) [14] defined as

$$H_S(\mathcal{G}_{v_U}^K)=-\sum_{u_U\in\mathcal{V}_{v_U}^K}P(u_U)\log P(u_U), \tag{4}$$

where $P(u_U)=D_{\mathcal{G}_{v_U}^K}(u_U,u_U)/\sum_{w_U\in\mathcal{V}_{v_U}^K}D_{\mathcal{G}_{v_U}^K}(w_U,w_U)$ is the probability of the SSRW visiting $u_U\in\mathcal{V}_{v_U}^K$, and $D_{\mathcal{G}_{v_U}^K}$ is the diagonal degree matrix of $\mathcal{G}_{v_U}^K$. □

The h-layer directed depth-based representation For $G(V,E)$ and its directed line graph $G_D(V_D,\overrightarrow{E}_D)$, we first establish the K-layer directed expansion subgraph of G_D through the undirected line graph $G_U(V_U,E_U)$ and the K-layer expansion subgraph $\mathcal{G}_{v_U}^K(\mathcal{V}_{v_U}^K;\mathcal{E}_{v_U}^K)$ on G_U. The K-layer directed expansion subgraph $\mathcal{G}_{v_D}^K(\mathcal{V}_{v_D}^K;\overrightarrow{\mathcal{E}}_{v_D}^K)$ around $v_D\in V_D$ is

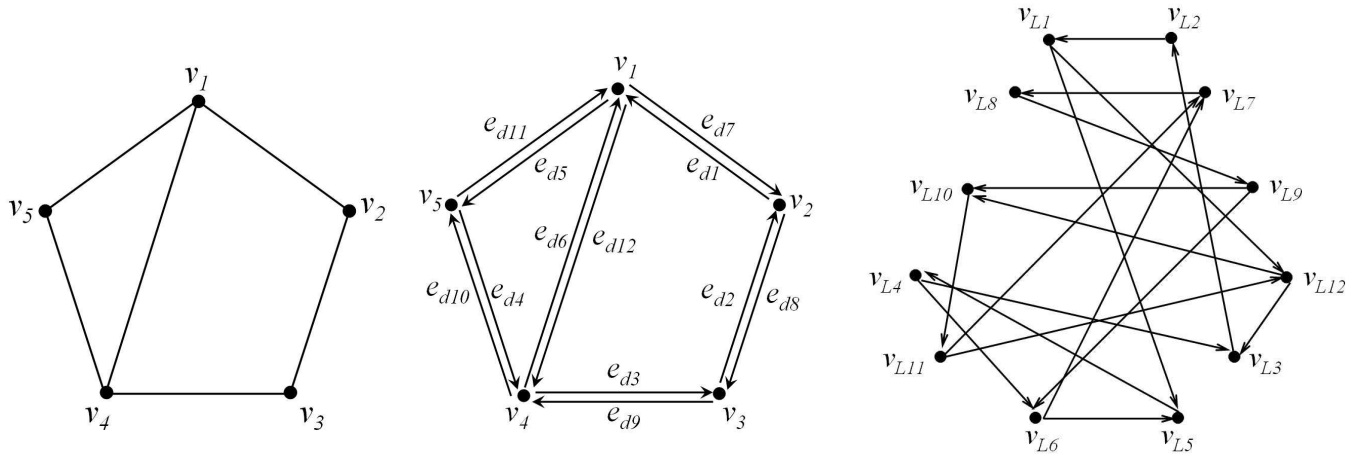
$$\left\{\begin{array}{l} \mathcal{V}_{v_D}^K \;\; =\{u_U\mid u_U\in\mathcal{V}_{v_U}^K\}; \\ \overrightarrow{\mathcal{E}}_{v_D}^K =\{\overrightarrow{(u_U,w_U)}\in\overrightarrow{E}_D\mid (u_U,w_U)\in\mathcal{E}_{v_U}^K\}. \end{array}\right. \tag{5}$$

where $v_D\in V_D$, $v_U\in V_U$, and $v_D=v_U$. In other words, the K-layer directed expansion subgraph $\mathcal{G}_{v_U}^K$ can be seen as a transformed graph of the K-layer expansion subgraph $\mathcal{G}_{v_U}^K(\mathcal{V}_{v_U}^K;\mathcal{E}_{v_U}^K)$ of G_U associated with the directed edges in G_D. For G_D, the h-layer directed DB representation around $v_D\in V_D$ is defined as

$$\overrightarrow{DB}_{G_D}^h(v_D)=[H_F(\mathcal{G}_{v_D}^1),\cdots,H_F(\mathcal{G}_{v_D}^K),\cdots,H_F(\mathcal{G}_{v_D}^h)]^\top, \tag{6}$$

where $H_F(\mathcal{G}_{v_D}^K)$ is the directed heat flow complexity measure of $\mathcal{G}_{v_D}^K$ [15], when the time t for computing the directed heat

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 17, 2016.



(a) Original graph (b) Digraph (c) Directed line graph

Fig. 1. Directed Line Graph Construction.

flow complexity H_F is infinity. H_F is defined as

$$H_F(\mathcal{G}^K_{v_D}) = F(\infty) = \frac{\overrightarrow{\mathcal{E}}^K_{v_D}}{\mathcal{V}^K_{v_D}}. \tag{7}$$

where $F(\infty)$ indicates that the time t for computing $H_F(\mathcal{G}^K_{v_D})$ is infinity. □

The h-layer depth-based representations of edges For the directed line graph $G_D(V_D, E_D)$ transformed from the original graph $G(V, E)$, each vertex represents a corresponding edge of $G(V, E)$. As a result, either the undirected depth-based representation of a vertex in $G_U(V_U, E_U)$ or the directed depth-based representation of a vertex in $G_D(V_D, \overrightarrow{E}_D)$ can be seen as a vectorial signature of a corresponding edge in $G(V, E)$. Thus, for the graph $G(V, E)$ and its directed line graph $G_D(V_D, \overrightarrow{E}_D)$, the h-layer depth-based representation of an edge $e \in E$ that is represented by a vertex $v_D \in V_D$ can be computed as

$$DB^h(v_D) = DB^h_{G_U}(v_U) + \overrightarrow{DB}^h_{G_D}(v_D), \tag{8}$$

where $v_U \in V_U$, $v_D \in V_D$, $v_U = v_D$ and $V_U = V_D$. $DB^h(v_D)$ can be seen as a mixed h-layer depth-based representation by summing both the h-layer undirected depth-based representation defined in Eq.(3) and the h-layer directed depth-based representation defined in Eq.(6). □

C. The Edge-based Matching Kernel through the Directed Line Graphs

In this section, we compute the edge-based matching kernel for graphs by aligning the vertices of their directed line graphs. Because, for an original graph and its directed line graph, each vertex of the line graph represents a corresponding edge in the original graph. For a pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$, $G_{D;p}(V_{D;p}, \overrightarrow{E}_{D;p})$ and $G_{D;q}(V_{D;q}, \overrightarrow{E}_{D;q})$ are their directed line graphs. For each directed line graph, we compute the h-layer depth-based representations as the vectorial signatures of its vertices. We compute the Euclidean distance between the depth-based representations $DB^h(v_i)$ and $DB^h(v_j)$ as the distance measure of the pairwise vertices v_i and u_j of the directed line graphs $G_{D;p}$ and $G_{D;q}$, respectively. The affinity matrix element $R(i, j)$ is defined as

$$R(i,j) = \sqrt{[DB^h(v_i) - DB^h(u_j)]^T[DB^h(v_i) - DB^h(u_j)]}. \tag{9}$$

where R is a $|V_{D;p}| \times |V_{D;q}|$ matrix. The element $R(i, j)$ represents the dissimilarity between the vertex v_i in $G_{D;p}$ and the vertex u_j in $G_{D;q}$. The rows of $R(i, j)$ index the vertices of $G_{D;p}$, and the columns index the vertices of $G_{D;q}$. If $R(i, j)$ is the smallest element both in row i and in column j, there should be a one-to-one correspondence between the vertex v_i of $G_{D;p}$ and the vertex u_j of $G_{D;q}$. We record the state of correspondence using the correspondence matrix $C \in \{0, 1\}^{|V_{D;p}||V_{D;q}|}$ satisfying

$$C(i,j) = \begin{cases} 1 & \text{if } R(i,j) \text{ is the smallest element} \\ & \text{both in row } i \text{ and in column } j; \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Eq.(10) implies that if $C(i, j) = 1$, the vertices v_i and v_j are matched. Since v_i and v_j represent a pair of corresponding edges in the original graphs G_p and G_q, the pair of edges can also be seen matched. As a result, based on the definition in [1], the edge-based matching kernel for the pair of original graphs G_p and G_q can be computed as

$$k^{(h)}_{EB}(G_p, G_q) = k^{(h)}_{EB}(G_{D;p}, G_{D;q}) = \sum_{i=1}^{|V_{D;p}|} \sum_{j=1}^{|V_{D;q}|} C(i,j). \tag{11}$$

which counts the number of matched vertex pairs between the directed line graphs $G_{D;p}$ and $G_{D;q}$.

D. Computational Complexity

For a pair of graphs each having n vertices, computing the edge-based matching kernel through their directed line graphs requires time complexity $O(n^6)$, for the worst case. This is because the matching kernel is based on the depth-based representations of the line graphs. Computing the representation

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 17, 2016.

from the line graphs relies on the computation of the shortest path matrix. The time complexity of computing the shortest matrix is cubic in the vertex number of the directed line graph. Thus, assume each directed line graph has m vertices, the time complexity of computing the edge-based matching kernel is $O(m^3)$. On the other hand, for each original graph, there may be at most $n(n-1)$ directed arcs residing on the edges and thus at most $n(n-1)$ vertices in the directed line graph, i.e., $m = n(n-1)$. As a result, computing the edge-based matching kernel through the line graphs may require time complexity $O(n^6)$, for the worst case.

The above computational analysis indicates that the edge-based matching kernel is not applicable for large graphs, since each of these graphs may have many edges. On the other hand, we can observe that the computational complexity of the matching kernel relates to the number of the edges. One way to overcome the computational inefficiency is to reduce the number of the edges. This in turn provides us a way of developing a new edge-based matching kernel through a sparser structure of the original graph. Clearly, the new kernel will be more efficient than the original kernel.

III. The Edge-based Matching Kernel on the Commute-time Spanning Trees

In this section, we proposed a new edge-based matching kernel on minimum spanning trees, based on the commute time. We show that the new kernel reduces the time complexity of the original edge-based matching kernel on original graphs from $O(n^6)$ to $O(n^3)$.

A. Graph Simplification from Commute Time

As we have stated in Section II, the original edge-based matching kernel [1] is computational expensive, and is not applicable. This follows the fact that there may be many edges in the original graphs. On way to overcome this problem is to reduce the number of edges. To this en, we propose to compute a sparser version of the original networks through the commute time [2].

Let $\mathbb{G}$ be the set of financial market networks that will be used in this work. These networks are all complete weighted graphs and have a fixed number of vertices. Let $G(V,E)$ be a sample graph/network from $\mathbb{G}$, with a weight function $\omega : V \times V \to \mathbb{R}^+$. If $\omega\{e(u,v)\} > 0$ ($\omega\{e(u,v)\} = \omega\{e(v,u)\}$), we refer to $e(u,v)$ as an edge of G, and we say that $u \in V$ and $v \in V$ are adjacent. Let A denote the adjacency matrix of $G(V,E)$, which satisfies $A(u,v) = w(u,v)$. The degree matrix D is the diagonal matrix with diagonal entries $D(u,u) = \sum_v A(u,v)$. Then, the graph Laplacian is given by $L = D - A$. The spectral decomposition of the Laplacian is $L = \Phi\Lambda\Phi^T$, where Φ is the $n \times n$ matrix $\Phi = (\phi_1|\phi_2|...|\phi_n)$ with the ordered eigenvectors as columns and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$ is the $n \times n$ diagonal matrix with the ordered eigenvalues as elements, such that $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.

The heat equation defines the dynamics of a diffusion process over the graph G. This is a partial differential equation associated with the graph Laplacian, i.e., $\frac{\partial \mathcal{H}_t}{\partial t} = -L\mathcal{H}_t$, where $\mathcal{H}_t$ is the heat kernel at time t. The solution of the heat equation is $\mathcal{H}_t = \exp(-Lt) = \Phi \exp(-\Lambda t)\Phi^T$. The evolution of a classical continuous-time random walk on G is also governed by the heat equation. Let p_t be the probability state vector of the walk, i.e., the vector whose components are the probabilities of the walk visiting the vertices of the graph at time t. Given an initial distribution p_0, the state vector of the walk at time t is $p_t = \mathcal{H}_t p_0$.

The hitting time $O(u,v)$ of a random walk is defined as the expected number of steps to go from vertex u to vertex v. The commute time is similarly defined as the expected number of steps to go from u to v, and then return to u. For the graph G, this is simply twice the hitting time. It can be shown that the hitting time can be written in terms of the eigendecomposition of the normalized Laplacian [16]

$$O(u,v) = \frac{1}{4}\sum_{j=2}^{n}\frac{1}{\lambda_j}(\phi_j(u) - \phi_j(v))^2, \tag{12}$$

where n is the number of vertices in G, i.e., $n = |V|$.

It is possible to simplify the structure of a graph by computing the minimum spanning tree over the commute time matrix [2]. This reduces the number of edges of the graph G to $n-1$, while retaining the characteristic structural information of the original graph. More specifically, for the complete weighted graph $G(V,E) \in \mathbb{G}$, we commence by computing the associated commute time matrix CT with entries

$$CT(u,v) = 2O(u,v). \tag{13}$$

With the commute time matrix to hand, we construct a new complete weighted graph $G_W(V_W, E_W)$ over the same vertex set of the original graph G, i.e., $V_W = V$. The weight of the edge between a pair of vertices in G_W is the commute time between the vertices of G. Using Kruskal's method [17], we can compute the minimum spanning tree $\mathcal{G}_S(\mathcal{V}_S, \mathcal{E}_S)$ over G_W. Note that, the spanning tree $\mathcal{G}_S$ has the same vertex set with G. The commute time is robust under the perturbation of graph/network structures. Thus, the minimum spanning tree $\mathcal{G}_S$ constructed on the commute time matrix can reflect the dominant structural information of the original graph G, while yielding a sparser structure.

B. The New Kernel on Spanning Trees

In this subsection, we define the new edge-based matching kernel on spanning trees. For a pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$, we commence by transforming them into the corresponding minimum spanning trees $\mathcal{G}_{p;S}(\mathcal{V}_{p;S}, \mathcal{E}_{p;S})$ and $\mathcal{G}_{q;S}(\mathcal{V}_{q;S}, \mathcal{E}_{q;S})$, respectively. The proposed kernel $k_{ST}^{(h)}$ for G_p and G_q is defined as

$$k_{ST}^{(h)}(G_p, G_q) = k_{EB}^{(h)}(G_{D;p}, G_{D;q}), \tag{14}$$

where $k_{EB}^{(h)}$ is computed as in Eq.(11).

Discussions As we have stated, the commute time captures the structural properties of the original graphs and is robust under perturbation of the graph structures. Thus, the minimum spanning tree constructed on the commute time matrix can as far as possible reflect the characteristic structural information of the original graph, while yielding a sparser structure. Note that, for a graph G having n vertices, the computation of the commute time matrix is based on the eigen-decomposition of the normalized Laplacian. Therefore, the graph simplification step (i.e., computing the minimum spanning tree $\mathcal{G}_S$) through

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 17, 2016.

the commute time requires time complexity $O(n^3)$. On the other hand, the minimum spanning tree $\mathcal{G}_S$ can reduce the edge number of G to $n-1$. Thus, there will be $2(n-1)$ vertices in the directed line graph that is transformed from $\mathcal{G}_S$. As a result, for a pair of graphs each having n vertices, computing the edge-based matching kernel $k_{ST}^{(h)}$ through the spanning trees requires $O(n^3)$, a considerable improvement from the original time complexity $O(n^6)$ of computing the edge-based matching kernel $k_{EB}^{(h)}$ on the original graphs.

IV. Experimental Results

A. Graph Datasets

We demonstrate the performance of our new kernel on five standard graph datasets from computer vision and bioinformatics databases. These datasets include BAR31, BSPHERE31, GEOD31, SHOCK, PPIs and CATH2. Details of these datasets can be found as follows.

BAR31: The SHREC 3D Shape database consists of 15 classes and 20 individuals per class, that is 300 shapes [18]. This is an usual benchmark in 3D shape recognition. From the SHREC 3D Shape database, we establish a graph datasets named BAR31 dataset through a mapping function. The function is ERG barycenter: distance from the center of mass/barycenter to the all other points.

Shock The Shock dataset consists of graphs from the Shock 2D shape database. Each graph is a skeletal-based representation of the differential structure of the boundary of a 2D shape. There are 150 graphs divided into 10 classes. Each class contains 15 graphs. The number of maximum, minimum and average vertices for the dataset are 33, 4 and 13.16 respectively.

PPIs: The PPIs dataset consists of protein-protein interaction networks (PPIs). The graphs describe the interaction relationships between histidine kinase in different species of bacteria. Histidine kinase is a key protein in the development of signal transduction. If two proteins have direct (physical) or indirect (functional) association, they are connected by an edge. There are 219 PPIs in this dataset and they are collected from 5 different kinds of bacteria with the following evolution order (from older to more recent) *Aquifex*4 and *thermotoga*4 PPIs from *Aquifex aelicus* and *Thermotoga maritima*, *Gram-Positive*52 PPIs from *Staphylococcus aureus*, *Cyanobacteria*73 PPIs from *Anabaena variabilis* and *Proteobacteria*40 PPIs from *Acidovorax avenae*. There is an additional class (*Acidobacteria*46 PPIs) which is more controversial in terms of the bacterial evolution since they were discovered. We select *Proteobacteria*40 PPIs and *Acidobacteria*46 PPIs as the second group test graphs. The maximum, minimum and average number of vertices of selected graphs are 232, 3 and 109.60 respectively.

CATH2: The CATH2 dataset has proteins in the same class (i.e., Mixed Alpha-Beta), architecture (i.e., Alpha-Beta Barrel), and topology (i.e., TIM Barrel), but in different homology classes (i.e., Aldolase vs. Glycosidases). **The CATH2 dataset is harder to classify, since the proteins in the same topology class are structurally similar**. The protein graphs are 10 times larger in size than chemical compounds, with $200-300$ vertices. There is 190 testing graphs in the CATH2 dataset.

B. Experiments on Graph Datasets

Experimental Setup: We evaluate the performance of our new edge-based matching kernel on commute-time spanning trees (EBSK), on graph classification problems. We also compare our kernel with several alternative state-of-the-art graph kernels. These graph kernels include 1) the previous edge-based matching kernel on original graphs [1], 2) the shortest path graph kernel (SPGK) [7], 3) the graphlet count graph kernel [19] with graphlet of size 4 (GCGK) [19], 4) the unaligned quantum Jensen-Shannon kernel (UQJS) [20]. For the EBMK kernel and the proposed EBSK kernel, we set the highest layer of the required depth-based representation as 10. The reason for this is that the 10-layer expansion subgraph rooted at a vertex of a directed line graph usually encapsulates most vertices of the line graph. For each kernel, we compute the kernel matrix on each graph dataset. We perform 10-fold cross-validation using the C-Support Vector Machine (C-SVM) Classification to compute the classification accuracy, using LIBSVM [21]. We use nine samples for training and one for testing. All the C-SVMs were performed along with their parameters optimized on each dataset. We report the average classification accuracy ($\pm$ standard error) and the runtime for each kernel in Table I and Table II, respectively. The runtime is measured under Matlab R2011a running on a 2.5GHz Intel 2-Core processor (i.e. i5-3210m).

TABLE II. Runtime of Computing the Kernel Matrix.

Datasets	EBMK	**EBSK**	SPGK	GCGK	UQJS
BAR31	302”	332”	11”	1”	630”
Shock	8”	10”	1”	1”	14”
PPIs	$> 2h$	597”	22”	4”	204”
CATH2	$> 2h$	3050”	253”	8”	4440”

Experimental Results: In terms of the classification accuracies, the new EBSK kernel outperforms or is competitive to any alternative graph kernel. Only the accuracies of the EBMK and GCGK kernels on the BAR31 and CATH2 datasets are a little higher than those of the new EBSK kernel. However, the EBSK kernel obviously outperforms these kernels on the PPIs and Shock datasets. The reasons of this effectiveness are twofold. First, unlike the SPGK, GCGK and UQJS kernels that ignore structural correspondence information between graphs, the new EBSK kernel can reflect correspondence information between edges of graphs. Second, the spanning trees for the EBSK kernel through the commute time can encapsulate dominant structural information of original graphs, relying on the nice properties of the commute time. In terms of the runtime, the new EBSK kernel is not the fastest kernel, but it can complete the computation in a polynomial time on any dataset. More significantly, the new EBSK kernel is more efficient than the EBMK kernel on the PPIs and CATH2 datasets that have large graphs with many edges. This is because the computational complexity of the EBSK and EBMK kernels relies on the edge number of original graphs, and the required spanning trees for the EBSK kernel can significantly reduce the edge number. On the other hand, the EBSK kernel is a little lower than the EBMK kernel on the BAR31 and Shock datasets. This is because the edge density of the two datasets is very sparse, reducing the edge number will not influence the computational efficiency of the EBMK kernel on original graphs. By contrast, the EBSK kernel on spanning trees needs extra time to compute the commute time matrix. These above

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 17, 2016.

TABLE I. CLASSIFICATION ACCURACY (IN $\%\pm$ STANDARD ERROR) USING C-SVM.

Datasets	EBMK	**EBSK**	SPGK	GCGK	UQJS
BAR31	**67.06** $\pm$ .64	62.30 $\pm$ .51	55.73 $\pm$ .44	23.40 $\pm$ .60	30.80 $\pm$ .61
Shock	42.06 $\pm$ .85	**42.13** $\pm$ 1.01	37.88 $\pm$ .93	27.06 $\pm$ .99	40.60 $\pm$ .92
PPIs	$--$	**72.47** $\pm$ .67	59.04 $\pm$ .44	46.61 $\pm$ .47	65.61 $\pm$ .77
CATH2	$--$	71.21 $\pm$ .63	32.57 $\pm$ .45	**73.68** $\pm$ 1.09	71.11 $\pm$ .88

observations demonstrate the effectiveness and efficiency of the new proposed EBSK kernel.

V. CONCLUSION AND FUTURE WORK

In this paper we have developed a new fast edge-based matching kernel on spanning trees through the commute time matrix. The spanning tree representation not only minimizes the edge number of the original graph but also preserves most of its structural information, relying on the nice properties of the commute time. Unlike the previous edge-based matching kernel on original graphs [1] that requires time complexity $O(n^6)$, the new matching kernel on spanning trees reduces the computational complexity to $O(n^3)$ that is a considerable improvement. We have evaluated the performance of the new proposed kernel on several standard graph datasets and demonstrate the effectiveness and efficiency.

In the future work, we will extend our analysis as follows. First, in our previous work [10] we have developed a hypergraph kernel based on subtree isomorphism test on directed line graphs. Similar to the edge-based matching kernel, the vertex set of the line graph also corresponds to the edge set of the original hypergraph. It would be interesting to use the commute time to develop a new hypergraph kernel on commute time spanning trees. Second, in our previous work [22], we have developed a novel framework for computing depth-based complexity traces of hypergraphs through the directed line graphs. It would be interesting to develop a new complexity trace method for hypergraphs through the commute time spanning trees.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant no.61503422), the Open Projects Program of National Laboratory of Pattern Recognition, and the program for innovation research in Central University of Finance and Economics. Edwin R. Hancock is supported by a Royal Society Wolfson Research Merit Award.

REFERENCES

[1] L. Bai, Z. Zhang, C. Wang, and E. R. Hancock, "An edge-based matching kernel for graphs through the directed line graphs," in *Proceedings of CAIP II*, 2015, pp. 85–95.

[2] H. Qiu and E. R. Hancock, "Graph simplification and matching using commute times," *Pattern Recognition*, vol. 40, no. 10, pp. 2874–2889, 2007.

[3] L. Bai, L. Rossi, H. Bunke, and E. R. Hancock, "Attributed graph kernels using the jensen-tsallis q-differences," in *Proceedings of ECML-PKDD*, 2014, pp. I:99–114.

[4] D. Haussler, "Convolution kernels on discrete structures," *Technical Report UCS-CRL-99-10, UC Santa Cruz*, 1999.

[5] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *ICML*, 2003, pp. 321–328.

[6] Z. Harchaoui and F. Bach, "Image classification with segmentation graph kernels," in *Proceedings of CVPR*, 2007.

[7] K. M. Borgwardt and H. Kriegel, "Shortest-path kernels on graphs," in *Proceedings of ICDM*, 2005, pp. 74–81.

[8] F. Aziz, R. C. Wilson, and E. R. Hancock, "Backtrackless walks on a graph," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24, no. 6, pp. 977–989, 2013.

[9] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, pp. 2539–2561, 2011.

[10] L. Bai, P. Ren, and E. R. Hancock, "A hypergraph kernel from isomorphism tests," in *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, 2014, pp. 3880–3885.

[11] F. R. Bach, "Graph kernels between point clouds," in *Proceedings of ICML*, 2008, pp. 25–32.

[12] L. Bai, P. Ren, X. Bai, and E. R. Hancock, "A graph kernel from the depth-based representation," in *Proceedings of S+SSPR*, 2014, pp. 1–11.

[13] P. Ren, R. C. Wilson, and E. R. Hancock, "Graph characterization via ihara coefficients," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 233–245, 2011.

[14] L. Bai and E. R. Hancock, "Graph kernels from the jensen-shannon divergence," *Journal of Mathematical Imaging and Vision*, vol. 47, no. 1-2, pp. 60–69, 2013.

[15] F. Escolano, E. Hancock, and M. Lozano, "Heat diffusion: Thermodynamic depth complexity of networks," *Physical Review E*, vol. 85, p. 206236, 2012.

[16] H. Qiu and E. R. Hancock, "Clustering and embedding using commute times," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1873–1890, 2007.

[17] R. C. Prim, "Shortest connection networks and some generalizations," *Bell system technical journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[18] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, and B. Falcidieno, "3d shape matching through topological structures," in *Proceedings of DGCI*, 2003, pp. 194–203.

[19] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," *Journal of Machine Learning Research*, vol. 5, pp. 488–495, 2009.

[20] L. Bai, L. Rossi, A. Torsello, and E. R. Hancock, "A quantum jensen-shannon graph kernel for unattributed graphs," *Pattern Recognition*, vol. 48, no. 2, pp. 344–355, 2015.

[21] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2011.

[22] L. Bai, F. Escolano, and E. R. Hancock, "Depth-based hypergraph complexity traces from directed line graphs," *Pattern Recognition*, vol. 54, pp. 229–240, 2016.

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 17, 2016.